PROCEEDINGS

55. IWK

Internationales Wissenschaftliches Kolloquium
International Scientific Colloquium

13 - 17 September 2010

Crossing Borders within the ABC

Automation,

Biomedical Engineering and

Computer Science

Faculty of
Computer Science and Automation

www.tu-ilmenau.de

TECHNISCHE UNIVERSITÄT
ILMENAU

Impressum
Published by

Publisher:	Rector of the Ilmenau University of Technology Univ.-Prof. Dr. rer. nat. habil. Dr. h. c. Prof. h. c. Peter Scharff
Editor:	Marketing Department (Phone: +49 3677 69-2520) Andrea Schneider (conferences@tu-ilmenau.de) Faculty of Computer Science and Automation (Phone: +49 3677 69-2860) Univ.-Prof. Dr.-Ing. habil. Jens Haueisen
Editorial Deadline:	20. August 2010
Implementation:	Ilmenau University of Technology Felix Böckelmann Philipp Schmidt

USB-Flash-Version.

Publishing House:	Verlag ISLE, Betriebsstätte des ISLE e.V. Werner-von-Siemens-Str. 16 98693 Ilmenau
Production:	CDA Datenträger Albrechts GmbH, 98529 Suhl/Albrechts
Order trough:	Marketing Department (+49 3677 69-2520) Andrea Schneider (conferences@tu-ilmenau.de)

ISBN: 978-3-938843-53-6 (USB-Flash Version)

Online-Version:

Publisher:	Universitätsbibliothek Ilmenau ilmedia Postfach 10 05 65 98684 Ilmenau

Home / Index:
http://www.db-thueringen.de/servlets/DocumentServlet?id=16739

DESIGNING AN APPLICATION FOR FIELD PROGRAMMABLE GATE ARRAYS – A CASE STUDY

Bernd Däne

Ilmenau University of Technology

ABSTRACT

This paper reports about a case study for designing an application for multiple programmable logic devices. It is based on a project with a decentralized communication system for a multiprocessor system. A fast serial communication system is implemented into one programmable logic device per processing node. Design tasks include system and data format design as well as structural synthesis. This case study focuses at evaluating a design flow that starts with high level modeling. Therefore the system was modeled using the *Matlab/Simulink* tool family and subsequently was transformed into logic design. Some qualitative and quantitative properties have been evaluated by simulation at high level and at low level, leading to some conclusions about practical use of such a design flow.

Index Terms – model based design, programmable logic device, multiprocessor communication, communication protocol, simulation

1. INTRODUCTION AND GOAL

Model based design should lead to an efficient design flow by enabling verification and validation at high abstraction levels and by automating important steps of synthesis of the actual implementation. Essential for this process is the availability and interoperability of appropriate design tools.

Since design support for programmable logic devices typically depends on manufacturer's proprietary tools here a design flow was used that combines such a tool with a commercial high level modeling tool.

Main goal of this case study is to practically evaluate this kind of design flow within a complex design project in order to get some conclusions about practical use of such a design flow.

2. PROJECT DETAILS

In the underlying project a multiprocessor system with a fast communication system is to be developed. This system has been presented in [1] and is successor of a former system that was used in [2] and [3]. It consists of three (and more in the future) processing nodes. Each node is equipped with a Digital Signal Processor (DSP) *TMS320C6713* of Texas Instrument's *C6000* DSP family [4].

The communication system is a fast serial interconnection structure using a unidirectional ring topology. In this way a low pin count and subsequently a compact structure is achieved, compared to former versions with parallel bus interconnection as in [5].

Communication is supported by programmable logic devices, more precisely by Field Programmable Gate Arrays (FPGA). Each DSP is connected to one FPGA by its local parallel bus. The FPGAs are *EP2C8-TQ208* devices of Altera's *Cyclone II* FPGA family [6].

The communication structure between the FPGAs uses Low Voltage Differential Signaling (LVDS, [7]) in order to get high throughput and sufficient robustness. One goal is maintaining the throughput that local parallel bus cycles of DSP devices can provide. Fig. 1 (from [8]) shows the general structure of the communication system.

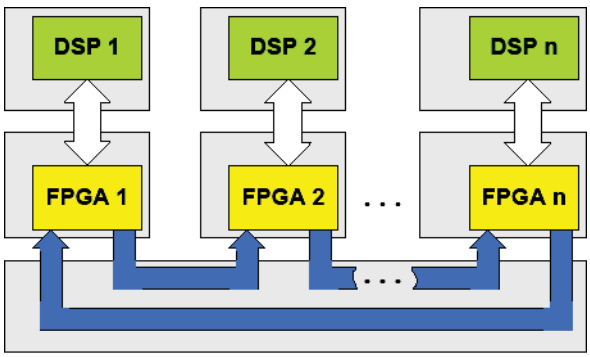

Fig. 1 General system structure

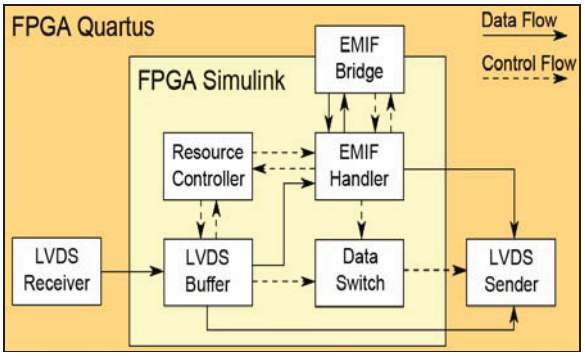

Fig. 2 Structure of one FPGA node

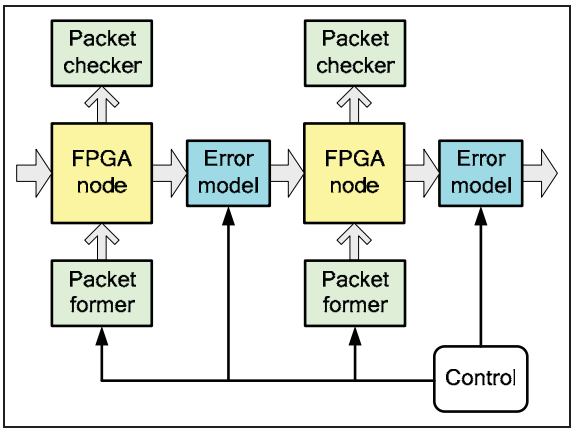

Fig. 3 Top level system

3. TOOLS AND DESIGN FLOW

For high level modeling the *Matlab/Simulink* tool with some add-ons has been used. Especially the *Simulink HDL Coder* add-on [9] exports VHDL source code [10] and therefore provides the link from *Matlab/Simulink* to hardware implementation.

Within *Matlab/Simulink* high level simulation support is available. For some parts of the model the *Stateflow* add-on has been used in order to model finite state machines (FSM).

Hardware synthesis has been done with Altera's *Quartus II* design tool [11]. This is the generic design tool for the FPGA device family mentioned above. It is able to import VHDL source code and to generate actual logic design.

Within *Quartus II* detailed simulation at lower abstraction levels is available, including estimate timing analysis. Furthermore there is some support for graphical modeling.

For additional validation Mentor Graphic's *ModelSim* (Altera Starter Edition, [12]) has been used. This tool provides extended simulation capabilities at VHDL level and proved useful for examining *Simulink HDL Coder* output before importing it into *Quartus II*.

4. HIGH LEVEL MODEL

The high level model for one FPGA node has been constructed using *Matlab/Simulink*. It includes modules for organizing the communication between the FPGAs (including error detection and correction functions) as well as communication buffers and control elements for interfacing to DSP's parallel bus (including control and status registers and interrupt functions). This paper will describe some selected parts of the model only.

4.1. Structure for one FPGA node

The structure of the model for one FPGA node is shown in Fig. 2 (from [13]). The term EMIF (External Memory Interface) here denotes the local parallel bus of the DSP [14].

As we see the LVDS circuitry has not been mod-

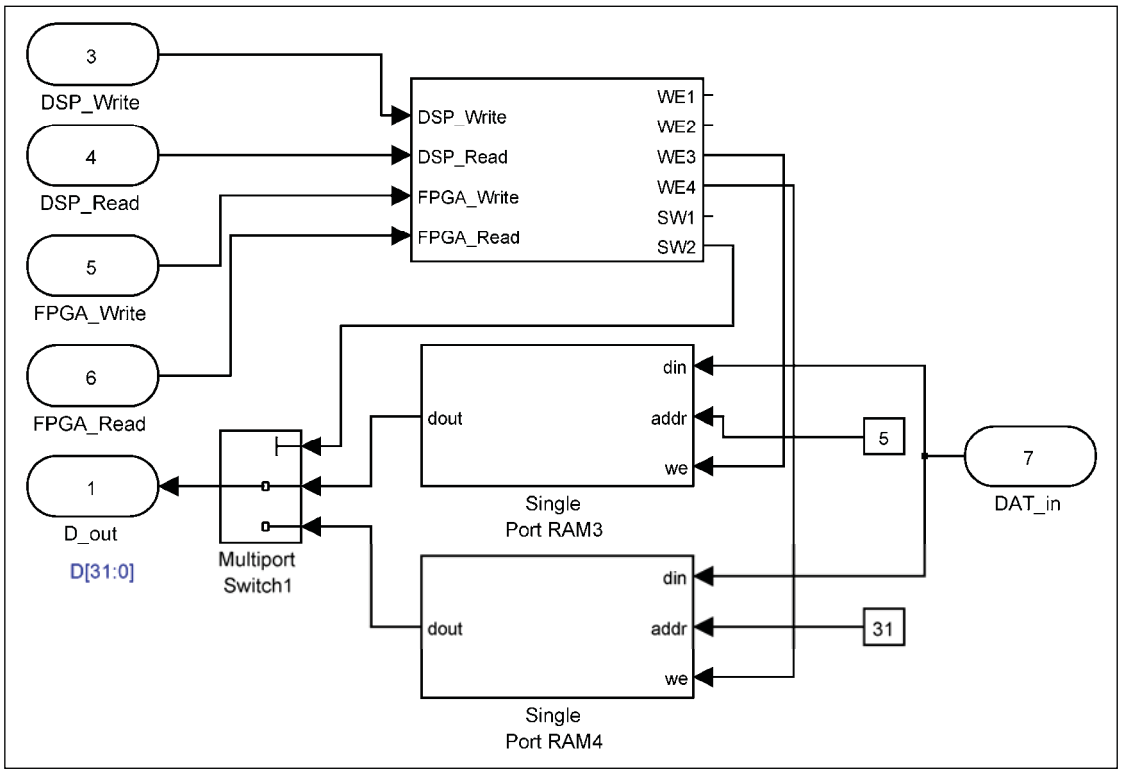

Fig. 4 Data buffer detail (from Matlab/Simulink model)

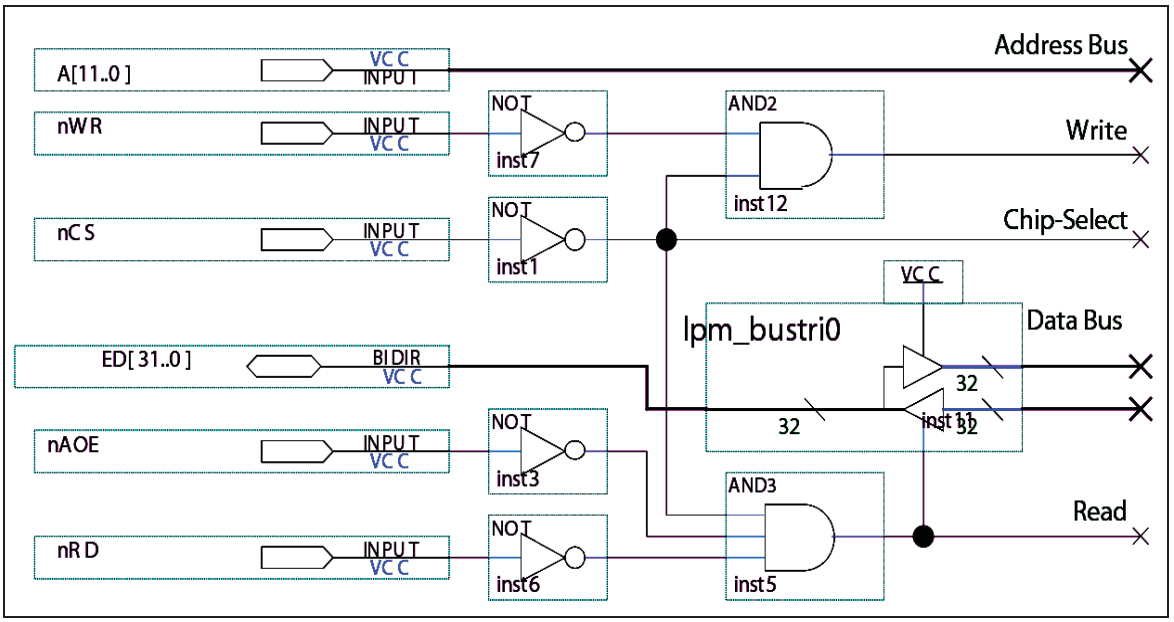

Fig. 5 FPGA to EMIF interface detail (from Quartus II model)

eled at high level. These parts later will be added in the *Quartus II* tool, when hardware synthesis is prepared. Since the LVDS connection transparently transports a serial data stream it may be omitted for high level simulation.

4.2. Simulation environment

For simulation within *Matlab/Simulink* a top level system of several such nodes has been build. In Fig. 3 (from [15]) its basic structure is shown. The FPGA nodes are connected according to the designated topology. An error model forces random bit-level errors in order to evaluate error detection and correction functions.

Packet formers (one per node) randomly generate data packets in order to be transmitted to other nodes. At the respective receiving node each of these packets will be checked for presence and correctness. This mechanism mimics the communication behavior of the DSPs and enables system evaluation under varying load conditions. A central control block coordinates some functions of simulation environment but not functions inside the FPGA nodes.

4.3. Other levels of model

Other levels of model are constructed either as regular *Simulink* block models or as FSMs using *Stateflow* functionality. As an example a small part of the "EMIF Bridge" is shown in Fig. 4 (from [8]).

We see a structure for buffered data transfer from communication system to DSPs EMIF bus interface. At the left there are control signals of DSP bus and of "EMIF Handler" (inside FPGA). Below them we see interfacing to the DSP's data bus. At the right there is a FPGA-internal data connection.

The upper block contains control logic (that is further modeled as FSM) while the others are two RAM blocks providing double buffering. During subsequent hardware synthesis these RAM blocks should be represented by predefined RAM blocks built into FPGA structure.

Please note that this figure is excerpted from a larger model sheet so that it apparently looks incomplete.

5. TRANSFORMING AND LOGIC SYNTHESIS

According to the planned design flow the FPGA design is to be exported as VHDL source code by the *Simulink HDL Coder* tool. This code will then be imported into the *Quartus II* tool in order to synthesize logic design for the FPGA device.

Characteristics and limitations of this step are of particular interest in this case study. As expected a fully automated transformation is not possible. Instead some special aspects have to be addressed.

5.1. Interfacing

As in every FPGA project, the design's interface to device pins has to be defined separately. Also the definition and attachment of some clock sources is required.

As an example Fig. 5 shows a detail excerpted from the *Quartus II* model (from [8]). It shows a part of the EMIF interface at FPGA pin level. The left side shows some signals of DSPs EMIF bus interface while signals at the right will connect to the core design that has been imported from the high level model.

As we see some simple signal qualification is done, such as inverting low-active signals (write sig-

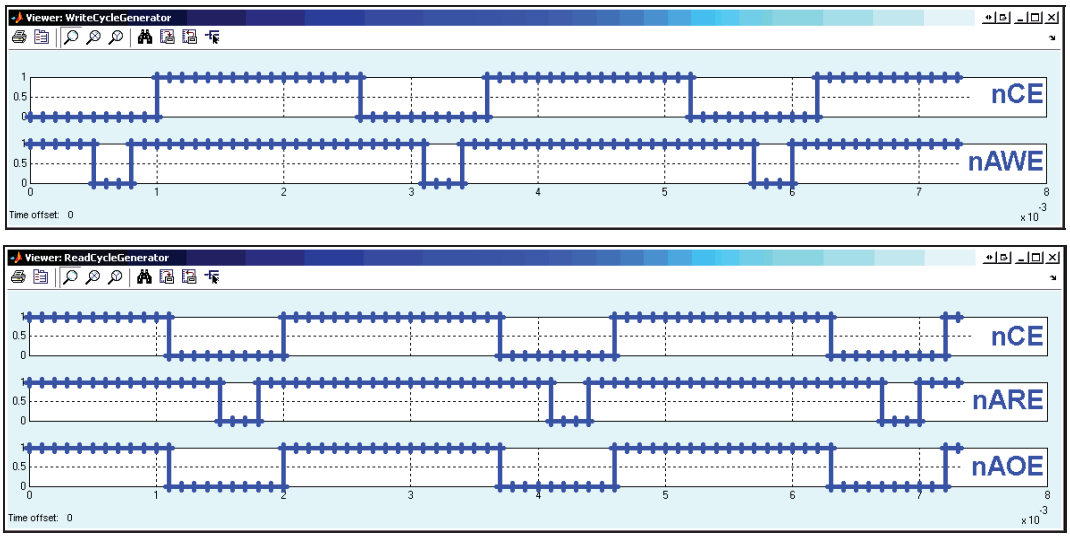

Fig. 6 Matlab/Simulink simulation for write cycles (top) and read cycles (bottom)

nal *nWR*, chip select signal *nCS*, address output enable *nAOE*, read signal *nRD*). Some of these signals are combined to form the inner *Read* and *Write* signals.

EMIF's bidirectional data bus *ED[31.0]* is split into two unidirectional busses because the high level model does not support bidirectional busses. This causes using of a driver unit and defining a control signal for bus direction.

In this modeling level furthermore the LVDS blocks have to be attached since they have no equivalent in the high level model. The LVDS blocks are predefined by FPGA manufacturer and should work transparently, meaning the data stream delivered by the LVDS receiver would be identical to the one that has been sent to the assigned LVDS transmitter.

But of course LVDS operation cannot be without latency. For this reason latency has been estimated experimentally and has been introduced into the high level *Matlab/Simulink* model as a delay block.

5.2. Module inference

Important features of modern FPGA devices are predefined modules, such as RAM blocks or multipliers that are built into FPGA structure. Such modules provide efficient implementation of these frequently used functions, which would be resource consuming if implemented by standard FPGA design elements.

Unfortunately, there is no direct mapping from *Matlab/Simulink* design elements to these blocks. Moreover such a mapping could restrict the high level model to one particular implementation platform.

So the low level design software has to detect such structures when analyzing designs that are imported from other tools. The *Quartus II* tool used in this project is equipped with such a mechanism. But not surprisingly in some cases it does not deliver the results expected.

In [8] some experiments are described that investigated how the *Quartus II* tool infers RAM blocks. It turned out that this ability depends on some details such as clock-gated control signal configuration of the memory blocks in the high level model.

This issue can be addressed by so-called 'control files' that modify the behavior of *Simulink HDL Coder* when generating VHDL code from the *Matlab/Simulink* model. For example these files may contain directives about clock properties of control signal interfaces.

When using appropriate control files module inference does work as expected. But this case shows that the design flow from high level model to implementation will not efficiently work without certain amount of human interaction and inspection.

6. SIMULATION

As mentioned above this case study also deals with simulation at different abstraction levels.

6.1. Simulation with high level model

For an efficient design process much validation work should be finished at high modeling level. So the simulation feature of the *Matlab/Simulink* tool is of special interest. Top level of the model under simulation was a three node structure as shown in Fig. 1, extended as shown in Fig. 3.

As a first example Fig. 6 (from [8]) shows timing diagrams for DSP bus cycles as generated by the packet former model component that mimics DSP behavior during simulation. We see some signals of the EMIF bus interface for write cycles (top) and read cycles (bottom). Goal of this test was to ensure that these signals will behave like the real EMIFs bus cycles [14].

Other simulation experiments investigated error detection and correction functions built into node

H = Header frame D = Data frame		Per hop count		Total
		1	2	
Frames total	H	16	14	30
	D	92	85	177
Frames with detected errors	H	0	0	0
	D	0	2	2
Frames with non-detected errors	H	0	0	0
	D	1	0	1

Table 1 Error figures for bit error rate 0.1%

design. At bit level a (31,5) Hamming code is implemented [13][16]. This code adds five extra bits to each block of 26 data bits. Additionally one ordinary parity bit is attached. This scheme provides correction of all one-bit errors and detection of all two-bit errors per block, as well as detection of 'stuck-at-1' and 'stuck-at-0' errors.

The error model as mentioned in Fig. 3 can generate stochastically distributed bit errors with an adjustable error rate as well as cyclic, clustered and 'stuck-at' errors [15]. With this setup the function of error detection and correction has been validated.

As an example Table 1 shows error figures for one simulation run with uniformly distributed bit errors at a bit error rate of 0.1% (from [15]). This rather high error rate provokes non-correctable and even non-detectable errors. Corrected errors are not shown in this table.

With 'hop count' the number of segments of travel is indicated. 'Two hops' indicate frames traveling through a transit node before reaching destination. Correctable errors will be corrected at each transit node as well as at destination node.

Frames divide into header frames containing routing and other protocol information, and data frames containing raw data. Note that non-corrected errors present an issue for higher protocol levels because information about frame type, sender or destination may be corrupted.

6.2. Simulation at lower levels

Simulation at VHDL level using *ModelSim* proved useful for fast checks of exported code before importing it into the *Quartus II* tool. Due to flexible test bench handling and some support from within *Simulink HDL Coder* (such as stimulus preparation) it leads much faster and easier to results then using the simulation feature of *Quartus II*. Most problems that arise from model transformation can be detected at this point [13][15].

Simulation within *Quartus II* remains necessary for validating interface structures that were added manually and for inspecting the results of module inference. Furthermore final timing analysis with respect to finished FPGA layout will be supported by this simulation level.

7. RESULTS AND CONCLUSION

In this paper a case study was presented that investigates a design flow leading from a high level model to a FPGA design. A number of tools has been examined, and some issues have been addressed that require human interaction during the process.

An important point was simulation at different abstraction levels providing ways for qualitative and quantitative validation of the design.

8. FURTHER WORK

Further development in this project will include simulation with higher node count and different communication topologies as well as implementation of higher levels of the communication protocol. At this point modeling the behavior of DSP software comes into range of sight. A hardware platform for implementing the design is in preparation. Development of the design workflow will address points of human interaction that need to be supported by rules, tools and reliable error avoidance strategies.

9. ACKNOWLEDGEMENT AND REMARKS

This work has been supported by German Research Foundation (Deutsche Forschungsgemeinschaft, DFG) under grant SFB 622.

10. REFERENCES

[1] F. Berger, B. Däne, A. Pacholik, W. Fengler, "Case Study: Performance Evaluation for Communication Strategies in Distributed Embedded Systems", The 2007 European Simulation and Modelling Conference (ESM'2007); St. Julian's, Malta, 22 - 24 October 2007, EUROSIS ETI Publication, 2007, pp. 582-585.

[2] B. Däne, “Modeling and Realization of DMA Based Serial Communication for a Multi Processor System”, Computer science meets automation: 52. IWK, Internationales Wissenschaftliches Kolloquium, 10 - 13 September 2007, proceedings, Vol. II, pp. 131-136, Ilmenau: Univ.-Verl., 2007.

[3] B. Däne, W. Fengler, “Modelling Hardware and Software for Fast Serial Interprocessor Communication”, DESDes'06 - Discrete Event System Design 2006, A Proceedings volume from the 3rd IFAC Workshop, Rydzyna, Poland, 26-28 September 2006, pp. 155-160, University of Zielona Góra Press, 2006.

[4] N. Dahnoun, “Digital Signal Processing Implementation Using the TMS320C6000 DSP Platform”, Prentice Hall, Harlow, 2000.

[5] B. Däne, F. Berger, “A Multiprocessor DSP System for a High Throughput Control Application”, EDERS-2004: The European DSP Education and Research Symposium, 16th November 2004, Birmingham, UK.

[6] Altera Corporation, “Cyclone II Device Handbook”, Volume I, Doc-ID: CII5V1-3.3., San Jose, Altera Corporation, February 2008.

[7] National Semiconductor Corp., “LVDS Owner's Manual”, 3rd Edition, Santa Clara, spring 2004.

[8] M. Müller, “Hardware und Software für Kommunikationsaufgaben in einem DSP-Multiprozessorsystem mit programmierbarer Logik” (Hardware and software for communication tasks in a multi DSP system with programmable logic devices), diploma thesis, Ilmenau Technical University, 2009.

[9] The MathWorks Incorporated, “Simulink HDL Coder 1”, User’s Guide, version 1.7, Natick, March 2010.

[10] P. J. Ashenden, “The designer's guide to VHDL”, 3rd edition, Elsevier B. V., New York, 2008.

[11] Altera Corporation, “Quartus II Handbook Version 10.0”, Volume I, Doc-ID: QII5V1-10.0.0, San Jose, July 2010.

[12] Altera Corporation, “Quartus II Handbook Version 10.0”, Volume III, Doc-ID: QII5V3-10.0, San Jose, July 2010.

[13] R. Schulze, “Modellbasierte Entwicklung der Interprozessorkommunikation des Multi-DSP-Kompaktsystems” (Model based development of inter processor communication functions for multi DSP compact system), bachelor thesis, Ilmenau Technical University, 2010.

[14] Texas Instruments, “TMS320C6000 DSP External Memory Interface (EMIF)”, user manual, Lit.-No. SPRU266E, Dallas, April 2008.

[15] E. Wolf, “Entwicklung und Realisierung einer FPGA-Anwendung für ein LVDS-Kommunikationssystem mit DSP-Ankopplung” (Development and implementation of a LVDS communication system with attached DSP), bachelor thesis, Ilmenau Technical University, 2010.

[16] T. K. Moon, “Error Correction Coding: Mathematical Methods and Algorithms”, Wiley Publishers, Hoboken, 2005.